INF3405

Hiver 2021

TP No. 1

Groupe 2

[1950276] – [Maxime Laroche]

[2028025] – [Zakaria Babahadji]

[2017113] - [Yun Ji Lao]

Soumis à : Bilal Itani.

26 Février 2021

Introduction

Le stockage de fichier sur sa machine n'est pas parfait. Un utilisateur peut vouloir accéder à ses fichiers lorsqu'ils n'est pas à côté de sa machine. Il peut aussi vouloir partager ses fichiers à quelqu'un d'autre. Pour répondre à ces besoins, le cloud existe. On dépose les fichiers désirés sur un serveur. Par la suite, le client peut avoir accès à ses fichiers partout dans le monde s'il sait communiquer avec le serveur.
Ce que l'on doit implémenter est un serveur capable de stocker des fichiers, ainsi qu'une ligne de commande pour qu'un client puisse communiquer avec le serveur.
Le client doit pouvoir voir les fichiers stockés sur le serveurs, se déplacer dans les répertoires, ainsi que télécharger et téléverser des documents sur sa machine.

Présentation de la solution

Lors du démarrage du serveur, la console demandera d'entrer un numéro de port et l'adresse à laquelle la communication entre le client et le serveur se fera. La méthode `askAdress`() a pour devoir de demander ces informations et elle s'assure que les informations entrées soient valides. L'adresse IP doit contenir 4 octet et doit être séparée par des points (ex: 127.0.0.1). Quant aux ports, il doit se situer entre 5000 et 5050. Toutes valeurs ne respectant pas ces contraintes ne seront pas acceptées. Lorsque toutes les informations sont entrées, un socket se crée du côté du serveur et du client avec ces informations. Le socket existe tout au long de la communication entre le serveur et le client. De plus, dans la command `upload`(), nous avons compris qu'il faut utiliser `socket.getOutputStream`() dans le côté de Client pour écrire dans le "Buffer"et socket.getInputStream() dans le côté de Serveur pour le recevoir. Il faut également vérifier l'existence du fichier avant le téléversement en servant de la fonction `.exists`(). La command `download`() est similaire à celle de upload, car le processus est simplement l'inverse de l'upload. Par contre, il fallait envoyer un socket de plus à Client pour signaler l' existence du fichier.

Ensuite, l'utilisateur entre différentes commandes lui permettant d'utiliser un répertoire. On a utilisé l'interface File pour implémenter ces méthodes. La première est `cd`(), qui permet de naviguer à travers des dossiers d'un répertoire. La seconde est la méthode `ls`(), qui permet d'afficher tous les fichiers et les dossiers d'un répertoire. La troisième est la méthode `mkdir` permettant de créer un nouveau dossier. Les deux dernières méthodes sont `upload`() et `download`() permettant de téléverser ou de télécharger un fichier dans le serveur. Lorsque l'utilisateur termine

s’est action il peut entrer la commande `exit` pour couper la communication avec le serveur.

Difficultés rencontrées

Ayant commencé le projet avant d’avoir vraiment compris le concept de socket, nous avons rencontré beaucoup de difficultés. Pour commencer, notre client créait un nouveau socket (en écrasant le vieux) au début de chaque requête. Cela génère des problèmes pour le serveur quant à la gestion des clients. Par la suite, nous avons utilisé des méthodes “toute faites” comme `InputStream.transferTo(OutputStream out)` pour faire le transfert de nos fichiers. Le transfert des fichiers s’effectuait correctement, mais rien n’indiquait à la personne qui reçoit quand arrêter de lire.
On a alors décidé d'envoyer la taille du fichier avant de l’envoyer, ainsi que d’envoyer ce dernier en petit paquets, plus facile à gérer.
Ensuite, étant novice au langage java, nous avons fait des erreurs lié à `«static»`. Une erreur apparaissait lorsque nous enlevions le keyword, alors on s’est dit qu’il fallait simplement le laisser. Nous avons appris par la suite qu’une variable static est une variable commune à toutes les instances de la classe et ce n’était pas ce que nous voulions. On a alors enlevé les `static` nécessaires.

Pour la commande cd, nous n'arrivons pas à implémenter le “..” pour se déplacer vers le répertoire parent. En effet, le problème est simplement que le “ new file “ créé n’a pas précisé son parent. Donc, il ne peut pas retourner au fichier parent. Pour régler ce problème, nous avons fait ajouter `currentDirectory` dans le premier argument du fichier. De cette façon, la nouvelle instance de fichier contient le pathname abstrait du parent. Finalement, la fonction `.transferto`() n’était pas idéale pour les sockets. Il permet de “copier “un fichier dans un autre fichier. Cette définition apparaît idéale, mais lorsqu’il lit un socket, il ne peut pas savoir la fin du fichier. Donc, il ne s'arrête jamais. Par conséquent, nous avons simplement abandonné cette fonction et la remplacer par un while boucle pour lire et écrire.

Critiques et améliorations :

Au début de ce projet, nous croyions que l'instruction était peu précise. Nous ne savons pas par ou commencer. Nous avons pu commencer par la validation de IP address et port grâce au directive de notre chargé de lab.

Conclusion

En conclusion, le but de ce projet est de faire une application serveur, et un client pour communiquer avec le serveur. Au cours de ce projet, nous avons renforcé la communication entre nous. En effet, malgré le confinement, nous avons fait plusieurs appels en dehors des cours pour discuter des concepts d'ingénierie en réseau informatique. Il nous a permis d'apprendre à coder une communication entre un client et un serveur. On a pu approfondir les notions vues en cours à ce sujet.

Il nous a aussi permis d'apprendre à implémenter un gestionnaire de fichier complet. De plus, au cours de ce laboratoire, nous avons appris la programmation en Java. En effet, nous avons régler plusieurs problèmes de débogage en Java grâce à la documentation de Oracle. La documentation a été très utile pour nous surtout pour l'implémentation de "`cd ..`". Bref, nos attentes ont été comblées, car notre application Client-Serveur fonctionne comme prévu. Donc, ma grande-mère peut stocker n'importe quel type de fichier dans notre serveur sans problème. On va être fier de nous après le don d'un vieux Pentium 3!